|  |
| --- |
|  |

|  |
| --- |
| **Technical Report** |
| ICTDBS507 – Integrate databases with websites |
|  |
|  |
|  |
| **Haris Khan** |
|  |
|  |

Table of Contents

[Part 1 – Analyse and Plan 2](#_Toc106978179)

[Summary of the project requirements 2](#_Toc106978180)

[Integration Plan 2](#_Toc106978181)

[Data Requirements 2](#_Toc106978182)

[The purpose of this integration project 2](#_Toc106978183)

[User Interaction 2](#_Toc106978184)

[The tools and technologies that will be utilised in this project 2](#_Toc106978185)

[The method of migrating existing data into the new system 2](#_Toc106978186)

[The intended methods of validating and sanitising data 2](#_Toc106978187)

[The strategy for testing and verifying the integration 2](#_Toc106978188)

[Email to manager requesting a review meeting 2](#_Toc106978189)

[Meeting Details and Feedback 3](#_Toc106978190)

[Actioned Meeting Feedback 3](#_Toc106978191)

[Requirements Checklist 3](#_Toc106978192)

[Part 2 – Database Integration 3](#_Toc106978193)

[Database Creation 3](#_Toc106978194)

[Connection String 3](#_Toc106978195)

[Data Access Mechanism 3](#_Toc106978196)

[Data Migration 4](#_Toc106978197)

[Web Application Connection 4](#_Toc106978198)

[Part 3 – User Interface Integration 4](#_Toc106978199)

[User Interface - Read 4](#_Toc106978200)

[User Interface – Modify 4](#_Toc106978201)

[User Interface – Formatting 4](#_Toc106978202)

[Validation and Sanitisation 4](#_Toc106978203)

[Testing 5](#_Toc106978204)

[Part 4 – Project Review 5](#_Toc106978205)

# Part 1 – Analyse and Plan

## Summary of the project requirements

To integrate a database with the website currently storing to text files. Migrate the data and convert all current functions to SQL database functions. Add new functionality for the website such as:

* Add new Products
* Update product prices
* Add an updated date to products
* Add a created date to any new comments
* Implementing a login screen to replace current system

## Integration Plan

### Data Requirements

* Display all products stored in a database
* Connect the existing functionality to a database:
  + Add new comments
  + Allow a user to modify their own recently created comments
  + Allow a user to delete their own recently created comments
  + View a list of all comments on a product

### The purpose of this integration project

*The purpose of the document is* to upgrade *Typical Tech Tools* to a functional database system to allow for streamlining all future updates to products, comments and the website. They are currently editing text files and it is not the best option in the long run for a growing company.

### User Interaction

*The user will interact by logging in, adding products(admin) or comments, editing or deleting their comments, updating prices, checking dates.*

### The tools and technologies that will be utilised in this project

*The tools used will be from SSMS SQL Server. Importing data tool from flat files. SQL queries and LINQ will be used. Dapper will also be used for the data access.*

### Technical Upgrades required for this project

Make sure SSMS is install and up-to-date and all related frameworks, package and dependencies are installed. Upgrade storage if database grows exponentially.

### The method of migrating existing data into the new system

Locating the csv files containing all data from the website. Importing csv files to SSMS and making any necessary changes if needed. Double check all created tables contain the correct corresponding values with csv files.

Using a Connection String data sourced to the SQL Client will be utilised to establish the database connection.

### The intended methods of validating and sanitising data

*- No negative values allowed for insertion*

- No non-numerical values for price

*- No empty fields before submission*

*- Currency in decimal format*

*- Only admin to create new products*

### The strategy for testing and verifying the integration

Test strategy is completed by performing test cases on the website in each requirement to make sure it working as intended. Such as creating a test case for adding a new product and confirming it is created in the SQL.

## Email to manager requesting a review meeting

To: Troy Vaughn

From: Haris Khan

CC:

BCC:

Date: 18/08/2022

Subject: Integration Plan

-------------------------------------------------------------

Hi,

I am emailing to request a meeting on this date: 20/08/2022 to confirm the integration plan for the website.

Kind Regards,

Haris Khan

----------------------------------------------

Haris Khan

HarisKhan@Software.com.au

## 

## Meeting Details and Feedback

|  |  |  |
| --- | --- | --- |
| **Date and Time of Meeting:** | 20/08/2022 |  |
| **Attendee names:** | Haris Khan | Troy Vaughn |
| **Review Feedback:** | | |
| * + Overall feedback has been positive, with minor clarifications with connection and tracking inputs | | |

## Actioned Meeting Feedback

*Record any changes made to the installation plan based on feedback from the review meeting*

* + Utilising dapper to connect to SQL
  + Use session data to track inputs

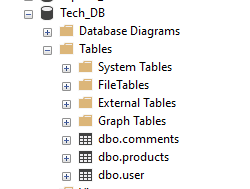
## Requirements Checklist

|  |  |
| --- | --- |
| **Completed** | **Requirement** |
|  | Integrate database (SQL) |
|  | Migrate data |
|  | Basic login page implemented |
|  | *Display all products* |
|  | *Convert existing functionality (comments)* |
|  | *Add new functionality (products)* |
|  | *Validate and sanitise inputs* |
|  |  |

# Part 2 – Database Integration

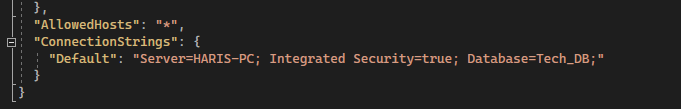
## Database Creation

*Screenshot from within an Integrated Development Environment or Relational Database Management System*

**

## Connection String

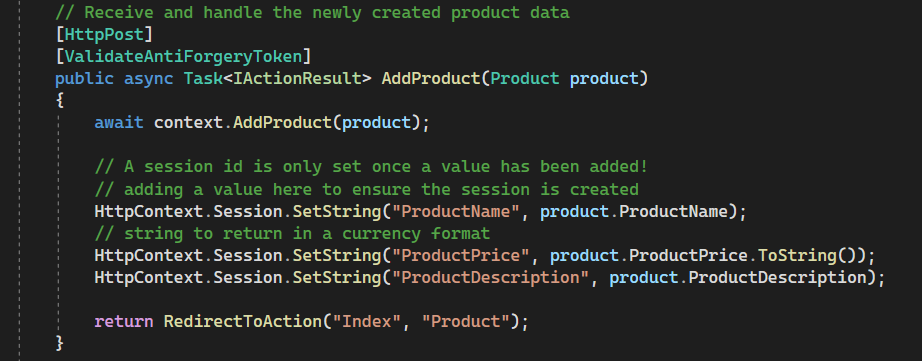
*Screenshot from within the relevant Integrated Development Environment.*

**

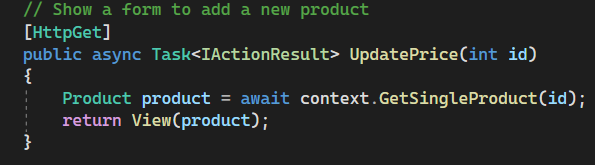
## Data Access Mechanism

*Screenshot of code demonstrating the CRUD methods that will interact with the database*

*Create*

**

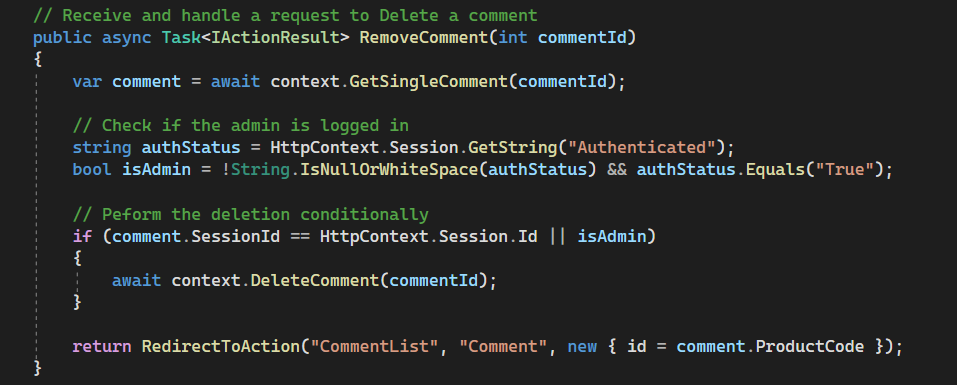
*Read*

**

*Update*

**

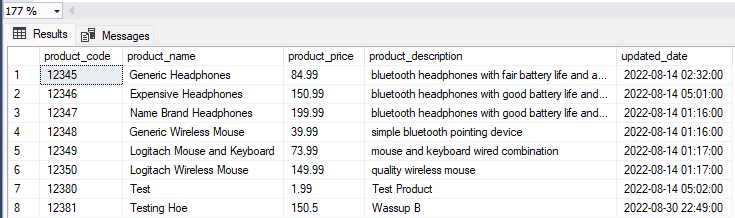
*Delete*

**

## 

## Data Migration

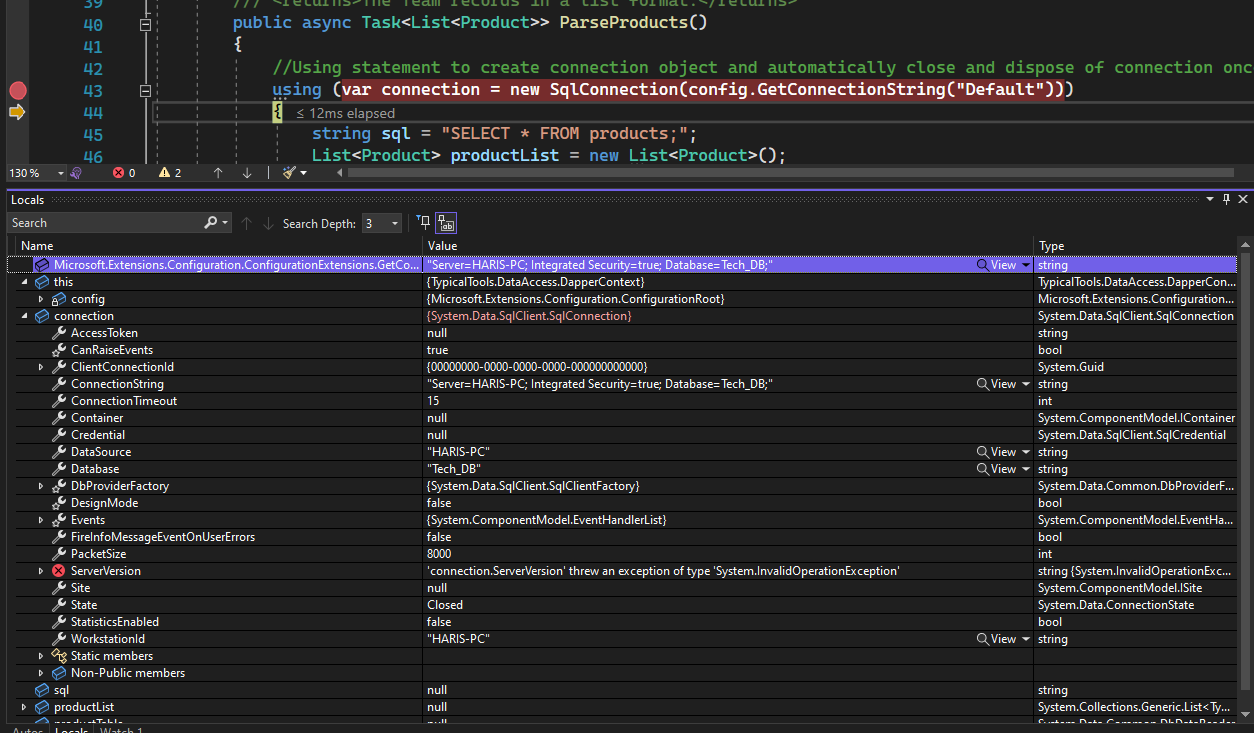
*Screenshot of the database tables showing the clients existing data migrated to the new system*

**

## 

## Web Application Connection

*Screenshot demonstrating a connection between the application and database, this can be as a breakpoint in code and does not need to involve the web applications user interface*

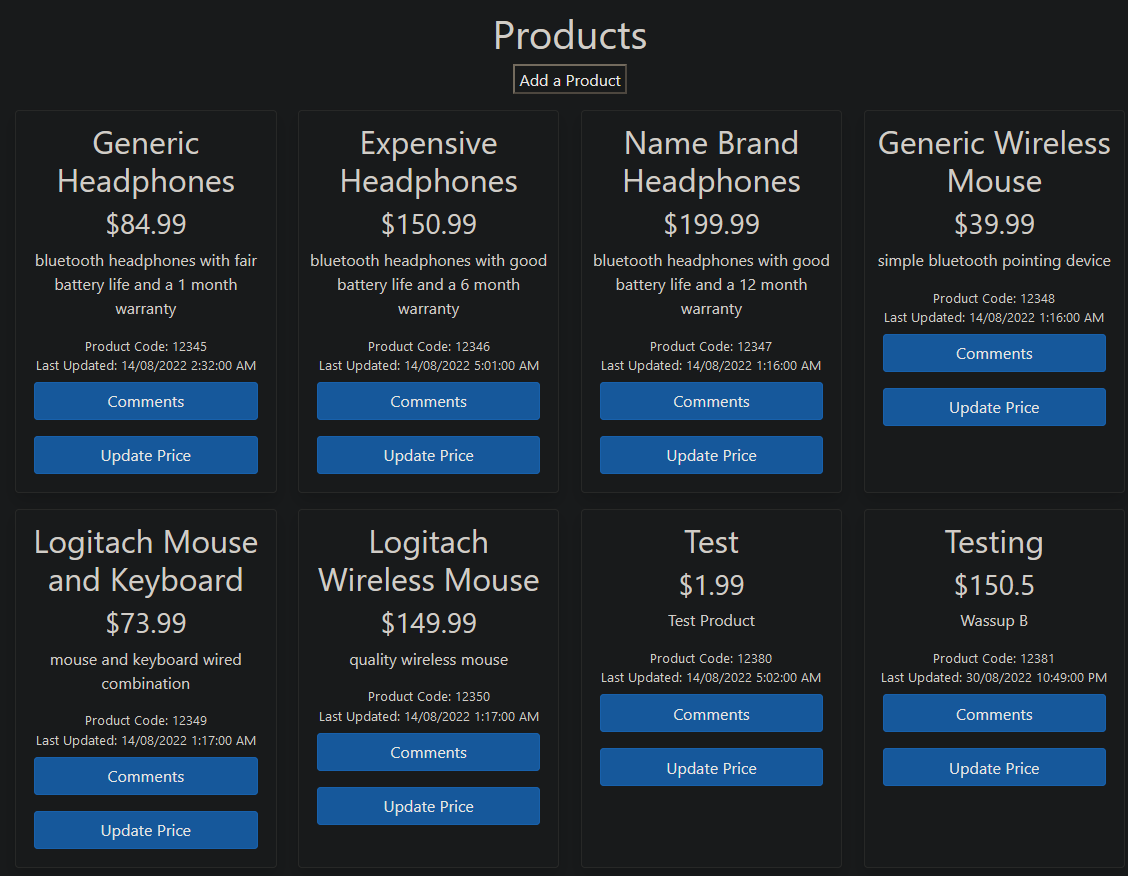
**

# 

# Part 3 – User Interface Integration

## User Interface - Read

*Provide a screenshot of data being read from the database and rendered in the user interface*

**

## User Interface – Modify

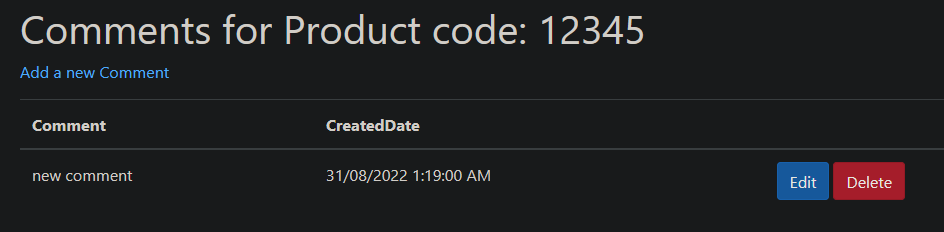
*Provide screenshots demonstrating the successful completion of each of these tasks, providing a before and after screenshot of each:*

* *Creating new Data in the Database (create page + resulting list page)*

*Before:*

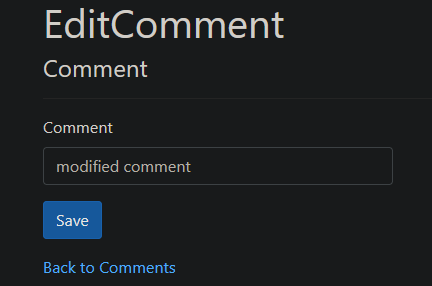
**

*After:*

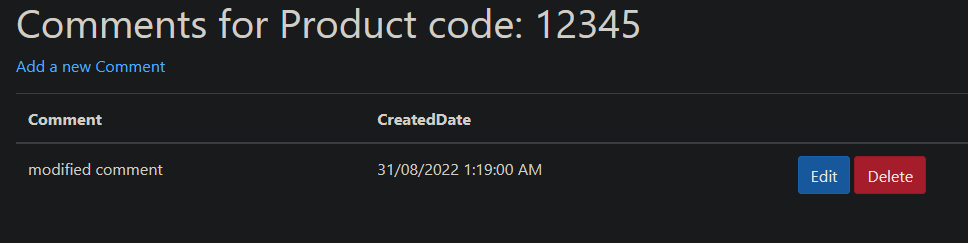
**

* *Modifying existing data (edit page + resulting list or details page)*

*Before:*

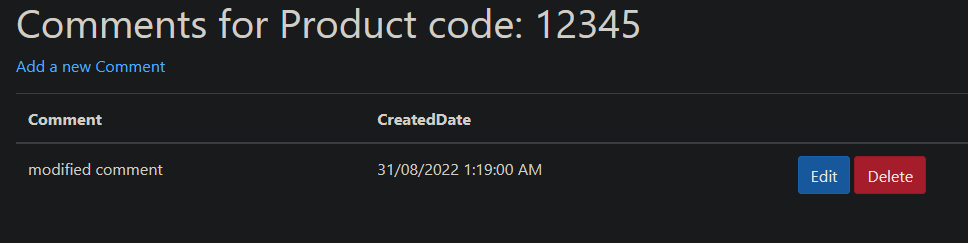
**

*After:*

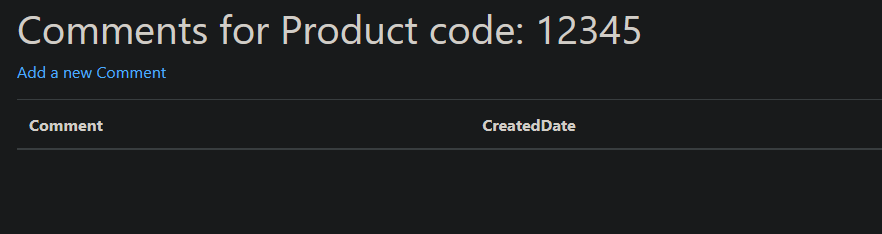
**

* *Deleting Data (before and after of list or details page)*

*Before:*

**

*After:*

**

## User Interface – Formatting

*Describe the organisational requirements that relate to the formatting of data, and how these have been met in this integration project.*

*The formatting of data has been met by formatting the prices to display only 2 decimal points instead of many digits as it was prior. The last updated date and comment created date have been added each specific product, the former being displayed right below the product code and the latter being displayed in the comment section. Also adding a ‘add product’ button for creation of products.*

## Validation and Sanitisation

*Describe how Validation and Sanitisation are being performed in this project, and the testing carried out to ensure that the validation and sanitisation are functional*

*Validation and Sanitisation are being performed in this project by ensuring no negative values are allowed for adding or updating a price. Another way is to make sure that there are no empty fields before submission of a product to ensure which ensures all fields are required. The prices format is now formatted to 2 decimal points instead of many digits as it was prior.*

*Testing to confirm these are performed by running the website and database, adding a new product, inserting negative values,* non-numerical values for price, *empty values, and many digits for products and checking to see if errors messages are working.*

## Testing

|  |  |
| --- | --- |
| **Test Performed** | **Testing results** |
| Running website on 2 different browsers (Firefox and Chrome) | Website is displaying and all functions are working as intended |
| Logging in as Admin User (Admin,Admin) and checking if login is authorised as true | Website loads products and add products is working for admins logins only |
| New product data is added into the website. Product name = TestProduct | Record TestProduct is saved accurately into database |
| Inserting negative values, non-numerical values for price, empty values, and many digits for new product (TestProduct) and checking to see if errors messages are working | Error messages are displayed:   * The field ProductPrice must be a number. * This field is required * Enter value with 2 decimals (eg. 45.00, 45.99) * Please enter a value greater than or equal to 0.01 |
| New comment data is added into the website. Comment name = TestComment | Record TestComment is saved accurately into database |
| Modifying TestProduct prices to see if last updated date is changed to DateTime.Now | Date successfully updates to current time |
| Verifying a created date is shown to current date time by inserting new comment | Date successfully created to current time |
| Checking sql database to see if comments are deleting from database after deletion | TestComment is deleted successfully from both website and database |
| Editing ‘TestComment’ to ‘ModifiedComment’ to confirm it is saving successfully | Comment altered successfully |

|  |  |
| --- | --- |
| **Completed** | **Requirement** |
|  | Integrate database (SQL) |
|  | Migrate data |
|  | Basic login page implemented |
|  | *Display all products* |
|  | *Convert existing functionality (comments)* |
|  | *Add new functionality (products)* |
|  | *Validate and sanitise inputs* |
|  |  |

# 

# Part 4 – Project Review

*Answer the following questions in relation to your project, providing full and professional responses.*

Describe at least 2 other database tools that could have been used in this project, including the pros and cons of those specific tools:

|  |
| --- |
| 1. Navicat for SQL Server:  * A graphical tool that could have been used in this database construction and management is called Navicat. With the help of this tool, creating and execution of SQL queries and instructions, create, update, and remove any database objects, and see data in tables would have been significantly easier. Its Cross-platform capabilities, Advanced SQL editor (code fragments, code autocompletion), and Object Designer are its main features. * Its main drawback is that the tool is a paid one and its capability for database and SQL server administration are unavailable. |
| 1. SQL Server Data Tool  * This tool could have been implemented in this project to assist in the designing in a Visual Studio environment. Its key positive attributes include its ability to operate with both a database project and a connected database instance, and its coverage of all database development stages. Reassembly, debugging, and database restructuring are all included. The Transact-SQL editor can also perform declarative changes. * The tool's limitations include the fact that it can only be used with Windows, that it requires Visual Studio to use, that it makes it difficult to write, update, and execute SQL queries, and that it is not appropriate for managing SQL Server. |

Describe how this project may have been approached or implemented differently if a NoSQL data provider was used in place of a SQL data provider:

|  |
| --- |
| In a NoSQL database, each document will be relating to a collection of products, and each document will contain JSON objects of the key-value: comments. Inside comments will contain the comment data. This means that there won’t be a need for 2 separate tables of Comments and Products as is the case in SQL. For example, if a product is deleted from NoSQL, it will delete all related comments as it is a JSON object of the collection, but in the case of SQL, the tables are separated and will need another trigger to delete the comments. |

Describe how statelessness factored into this integration project, what, if any steps had to be taken to work with or around the statelessness of your project.

|  |
| --- |
| Statelessness factored into this project by reducing the chances of a having the website ‘break’ because each request is independent and is its own resource. Due to the fact that in this project, the database is small and the connection load is minimal, no steps were required to work with or around the statelessness of the project. |

Describe the role of session management in your project, or the role it could have played in the implementation of this integration project.

|  |
| --- |
| The role of session management in this project is to store data (in variables) for usage on several pages at once. For example, session data of a new created comment. The data is not saved on the user's machine, unlike cookies. The value is kept on the server and because it keeps the data as client-based, a session is among the finest methods for state management. |

Describe an issue encountered in the integration and development process, including the process used to solve the issue.

|  |
| --- |
| One issue encountered during integration and development was getting Dapper to throw an exception ”System.Data.SqlClient.SqlConnection” after updating System.Data.SqlClient version. This was solved by NuGet package manager and searching for "System.Data.SqlClient" and installing the latest and the exception goes away. |